The Theory of Parametricity in Lambda Cube (Towards new interaction between category theory and proof theory)

AUTHOR(S):

Takeuti, Izumi

CITATION:

Takeuti, Izumi. The Theory of Parametricity in Lambda Cube (Towards new interaction between category theory and proof theory). 数理解析研究所講究録 2001, 1217: 143-157

ISSUE DATE:

2001-06

URL:

http://hdl.handle.net/2433/41237

数理解析研究所講究録 1217 巻 2001 年 143-157

The Theory of Parametricity in Lambda Cube

Takeuti Izumi 竹内泉

Graduate School of Informatics, Kyoto Univ., 606-8501, JAPAN
takeuti@kuis.kyoto-u.ac.jp
京都大学情報学研究科

Abstract. This paper defines the theories of parametricity for all system of lambda cube, and shows its consistency. These theories are defined by the axiom sets in the formal theories. These theories prove various important semantical properties in the formal systems. Especially, the theory for a system of lambda cube proves some kind of adjoint functor theorem internally.

1 Introduction

1.1 Basic Motivation

In the studies of informatics, it is important to construct new data types and find out the properties of such data types. For example, let T and T' be data types which is already known. Then we can construct a new data type $T \times T'$ which is the direct product of T and T'. We have functions *left*, *right* and *pair* which satisfy the following equations:

$left(pair\, xy) = x$, $right(pair\, xy) = y$ for any $x : T$ and $y : T'$,
$pair(left\, z)(right\, z) = z$ for any $z : T \times T'$.

As for another example, we can construct $List(T)$ which is a type of lists whose components are elements of T. We have the empty list $()$ as the elements of $List(T)$, the functions $(-,-)$, which is so called sons-pair, and *listrec*. These element and functions satisfy the following equations:

$listrec()fe = e$, $listrec(x,l)fe = fx(listrec\, lfe)$

for any $x : T$, $l : List(T)$, $e : D$ and $f : T \to D \to D$, where D is another type. Moreover, we have list induction such that:

Suppose that if $P(l)$ then $P((x,l))$ for any $x : T$ and $l : List(T)$.
Then, if $P(())$ then $P(l)$ for each $l : List(T)$,

where $P(\,)$ is an arbitrary property over $List(T)$.

Polymorphic type theory is very powerful for describing new data types. One of the most famous polymorphic type theory is System F. Although we have only two type constructor $\to$ and Π in System F, we can encode the direct product $T \times T'$ as $\Pi X.(T \to T' \to X) \to X$ in System F. And also we can encode the functions *left*, *right* and *pair* as some terms in System F. Similarly, the type of the lists is encodes as $\Pi X.(T \to X \to X) \to X \to X$, and the elements and functions $()$, $(-,-)$ and *listrec* are also encodable in System F. Under this encoding, we have the following equations up to beta-equivalence:

$left(pair\, xy) =_\beta x$, $right(pair\, xy) =_\beta y$
$listrec()fe =_\beta e$, $listrec(x,l)fe =_\beta fx(listrec\, lfe)$

But, the term $pair(left\ z)(right\ z)$ is not beta-equivalent to z. Moreover, the statement of list induction is not described in System F, because System F is a system for construction and calculation of terms with type assignment. Thus, we cannot handle the equation $pair(left\ z)(right\ z) = z$ nor list induction with System F. We call such properties semantic properties.

Under such encoding, we have the problem that semantical properties are not derivable. In order to avoid this problem, we sometimes use models of data types which satisfy the semiantical properties, such as the equation $pair(left\ z)(right\ z) = z$ or list induction. Such models are often constructed in category theory or domain theory. But it is more useful to solve that problem only in type theory itself with a formal way.

Our aim is to make formal logical system over terms of a type theory, and to derive semiantical properties by the logical system. A successful example is the formal theory of parametricity over terms of System F, which is studied in some literatures [ACC, PA, T]. The theory of parametricity proves many semantical properties, such as the equation $pair(left\ z)(right\ z) = z$ or list induction. Therefore, we would like to extend the theory of parametricity into more strong or general type theory, that is, lambda cube.

1.2 Interpretation of Category Theory Into Type Theory

Although it is successful to describe and prove some semantical properties by the theory of parametricity over System F, there are much more useful notions and semantical properties which cannot be described in System F. Some of them are functors, natural transformations, and adjunction, which are provided by category theory.

In order to describe such notions which come from category theory in type theory, we have to describe the basic notions of category theory, such as objects, hom-sets and arrows, in type theory. By comparing the parametric models of System F in the literatures [Hasegawa1, Hasegawa2, BAC] with the formal theories of parametricity in the literatures [ACC, PA, T], we find out the interpretation of category theory into type theory as following:

Category theory		Type theory
Object	$\longrightarrow$	Type
Exponential object, Hom-set	$\longrightarrow$	Functional type
Arrow	$\longrightarrow$	Element of the type
Limit	$\longrightarrow$	Universal type

This interpretation is very attractive. Therefore, we preserve this interpretation in extending our theory.

According to the interpretation above, there exists a category which is interpreted as the collection of all types. Here, we write $*$ for the collection of all types, as is done by Barendregt [Barendregt], and we write Ω for such category interpreted as $*$, as is done by Hasegawa [Hasegawa3]. Then, the functors of

Ω into Ω should be interpreted as a term of type $* \to *$. System F does not have the object of type $* \to *$, thus the functors are not internal objects in the formal theories of type theory in the literatures [ACC, PA, T]. On the other hand, the system $\lambda\omega$ in lambda cube has such objects of type $* \to *$, namely, $\lambda X : *. T \to X : * \to *$.

Our motivation is to extend the theory of parametricity to a formal logic which includes $\lambda\omega$, and prove some semantical properties on functors. For example, we will show the adjoint functor theorem for functors of $\Omega \to \Omega$, that is, if a functor of $\Omega \to \Omega$ preserves some kind of limits, then it has a left adjoint.

1.3 Lambda Cube

Lambda cube is studied by Barendregt [Barendregt]. It is a family of eight type assignment systems, which are $\lambda\!\to$, λP, $\lambda 2$, λP2, $\lambda\underline{\omega}$, $\lambda\text{P}\underline{\omega}$, $\lambda\omega$ and $\lambda\text{P}\omega$. The system $\lambda 2$ is equivalent to System F, and $\lambda\text{P}\omega$ is equivalent to the system of calculus of construction. We define the theory of parametricity for each system of lambda cube by using conformity relation.

The definitions for eight systems are uniform. That is only the reason that we define the theory of the parametricity for all of eight systems, nevertheless the theory of parametricity for $\lambda\!\to$, $\lambda\underline{\omega}$ or $\lambda\text{P}\underline{\omega}$ is not so important.

We use λP2 as predicate logic for $\lambda\!\to$, λP, $\lambda 2$ and λP2, and we use $\lambda\text{P}\omega$ as predicate logic for $\lambda\underline{\omega}$, $\lambda\text{P}\underline{\omega}$, $\lambda\omega$ and $\lambda\text{P}\omega$.

The systems λP2 and $\lambda\text{P}\omega$ seem stronger than what is necessary as predicate logic over $\lambda 2$ and $\lambda\omega$, because terms and proofs are not separated in the systems. Actually, the predicate logic for System F in precedent works [PA, T] is a subsystem of λP2. Barendregt also provides other family of systems named logic cube, in which terms and proofs are separated. But they have more constants and rules than λP2 and $\lambda\text{P}\omega$. We use λP2 and $\lambda\text{P}\omega$ because of the simplicity of rules. We can separate terms and proofs in the discussion of this paper easily, and it makes no problem.

The system λP with equality has a power enough to play the role of predicate logic for λP. But the rules of equality destroys our uniformity. Therefore, we use λP2 as predicate logic for λP, although λP2 is much stronger than what is necessary.

1.4 Related Works

The original meaning of parametricity was as a notion for models of polymorphic calculi, such as System F, which is second order lambda calculus studied by Girard [Girard]. More recently, many researchers have had interest in parametricity in formal logic. In this paper we also discusses parametricity in formal logic.

In the sense of Reynolds [Reynolds], parametricity is the property that if f is of universal type $\Pi X.F[X]$ then $fT(F[R])fU$ holds for all types T and U and for all relations R between the domains of T and of U. We regards the

parametricity notion as extended into the notion for all types, just as Plotkin and Abadi [PA] do.

Abadi, Cardelli and Curien [ACC] propose System R, a formal system for parametricity. It is a logical system for binary relations between terms of System F. Because it deals only with binary predicates, its expressive power is rather weak.

Hasegawa [Hasegawa1, Hasegawa2, Hasegawa3] makes a categorical model for polymorphism. If it is parametric, then it is a categorical model for System R. By formalising his informal logic in [Hasegawa1, Hasegawa2], we can obtain a formal treatment of parametricity in formal second order predicate logic, Such formal treatment is equivalent to the treatment in [PA, T]. Hasegawa [Hasegawa1, Hasegawa2] does not show the existence of the relation-frame which is parametric for all the types of System F, although he shows the existence of the relation-frame which is parametric for some particular types, such as pair types, sum types, natural numbers, and so forth.

Bellucci, Abadi, and Curien [BAC] construct a model of System R from a partial equivalence relation model of System F. This is a model theoretic proof of the consistency of the theory of parametricity.

Plotkin and Abadi [PA] formalise parametricity based on a second order predicate logic. In their system, the basic logic is not specialised to the treatment of parametricity, and the axioms implement the parametricity. Takeuti [T] gives a syntactic proof of the consistency of their system.

As a formal logical systems for dealing with semantic properties, Pfenning and Paulin-Morling [PP] proposed calculus of construction, which is equivalent to $\lambda P\omega$. They showed that calculus of construction can describe many semiantic properties. The set of axioms which we will give in this paper proves such semiantic properties.

There are some works to formalise category theory in type theory. One of them is by Huet et al [HS]. They work is to formalise category theory in Coq. In their work, both objects and arrows in category theory are interpreted as ground level object, or terms, in type theory, and Hom-set is interpreted as a ternary relation of the arrow, the domain and the codomain. That is directed to another direction than our interpretation, and therefore we cannot apply theory of parametricity for their work.

1.5 Outline

Section 2 has the brief introduction of lambda cube and the definitions of our notations. Section 3 has the definitions of conformity relation and of the axioms of parametricity. Section 4 has our interpretation of category theory into lambda cube. Especially, the adjunction theorem is proved in Section 4. Section 5 has concluding discussion.

2 Lambda Cube

2.1 Lambda Cube as a Type Assignment System

Lambda Cube is a family of eight systems studied by Barendregt. We will give the definition of them. We use the term 'phrase' instead of 'term' or 'expression'. First of all we will define the notions of prephrases and prebases.

Definition 2.1.1 (Prephrase) A syntactic class of *prephrases* are defined by the following syntax:

$$M ::= Var \mid \Box \mid * \mid \lambda Var : M.M \mid MM \mid \Pi Var : M.M$$

where *Var* is a *variable*. There are of infinitely many variables. Parentheses are supplied to parse, as $(\lambda x{:}M.M')(M''M''')$. The symbols λ and Π bind variables. The notions of *free variables*, *α-equivalence*, *substitution of variables*, and *β-reduction* are defined in the ordinary manner. We identify α-equivalent prephrases. We write $FV(M)$ for the set of all the free variables of M. We write $M[M'/x]$ for the prephrase given by substitution of x with M' in M.

Definition 2.1.2 (Prebasis) A prebasis is a finite sequence of expressions $x : M$ in which x is a variable and M is a prephrase which satisfies the following conditions. A sequence $\Gamma \equiv x_1 : M_1, x_2 : M_2, ..., x_n : M_n$ is a prebasis iff for each i,

$$x_i \notin \{x_1, x_2, ..., x_{i-1}\} \cup \bigcup_{1 \leqq j \leqq i} FV(M_j)$$

The set $\{x_1, x_2, ..., x_n\}$ is called the domain of Γ, and written as $dom(\Gamma)$. The length n may be 0, thus, an empty sequence is also a prebasis. An expression $x : M$ is called a *clause* in a prebasis. A prebasis Γ is equivalent to another prebasis Γ' iff Γ is a permutation of Γ'. We write $\Gamma \simeq \Gamma'$ when Γ is equivalent of Γ'. Note that $\Gamma \simeq \Gamma'$ only if both Γ and Γ' are prebases.

Definition 2.1.3 (Subsequence) Let Γ and Γ' be two prebases. The relation $\Gamma \underline{\ll} \Gamma'$ holds iff there is a sequence $i_1, i_2, ..., i_n$ such that:

$$1 \leqq i_1 < i2 < \cdots < i_n \leqq m$$
$$\Gamma \equiv x_{i_1} : M_{i_1}, x_{i_2} : M_{i_2}, ..., x_{i_n} : M_{i_n}$$
$$\Gamma' \equiv x_1 : M_1, x_2 : M_2, ..., x_m : M_m$$

We write $\Gamma \leq \Gamma'$ and say that Γ is a *subsequence* of Γ' when there is a prebasis Γ'' such that $\Gamma \underline{\ll} \Gamma''$ and $\Gamma'' \simeq \Gamma'$. It is easily seen that $\leq$ is a reflexive transitive relation, and if $\Gamma \leq \Gamma'$ and $\Gamma \geq \Gamma'$ then $\Gamma \simeq \Gamma'$.

Definition 2.1.4 (Marge) Let Γ, Γ' and Γ'' be three prebases. The basis Γ is a *merge* of Γ' and Γ'' iff $\Gamma \leq \Gamma'$, $\Gamma \leq \Gamma''$ and each clause $x : M$ in Γ appears either in Γ' or in Γ''.

Definition 2.1.5 (Lambda Cube) Let *Cube* be a set of symbols such that $Cube = \{\rightarrow, \mathrm{P}, 2, \mathrm{P2}, \underline{\omega}, \mathrm{P}\underline{\omega}, \omega, \mathrm{P}\omega\}$. A set $\lambda Cube$ consists of λS for all $S \in$

Cube. A map $|\cdot|$ is an injection of *Cube* into the power set of $\{*, \Box\}^2$, which is defined as below.

$$\begin{array}{rcl}
|\to| & = & \{\ \langle *,*\rangle \qquad\qquad\qquad\qquad\ \} \\
|\mathrm{P}| & = & \{\ \langle *,*\rangle, \langle *,\Box\rangle \qquad\qquad\ \} \\
|2| & = & \{\ \langle *,*\rangle, \qquad \langle \Box,*\rangle \qquad\ \} \\
|\mathrm{P2}| & = & \{\ \langle *,*\rangle, \langle *,\Box\rangle, \langle \Box,*\rangle \qquad\ \} \\
|\underline{\omega}| & = & \{\ \langle *,*\rangle, \qquad\qquad\ \langle \Box,\Box\rangle\ \} \\
|\mathrm{P}\underline{\omega}| & = & \{\ \langle *,*\rangle, \langle *,\Box\rangle, \qquad \langle \Box,\Box\rangle\ \} \\
|\omega| & = & \{\ \langle *,*\rangle, \qquad \langle \Box,*\rangle, \langle \Box,\Box\rangle\ \} \\
|\mathrm{P}\omega| & = & \{\ \langle *,*\rangle, \langle *,\Box\rangle, \langle \Box,*\rangle, \langle \Box,\Box\rangle\ \}
\end{array}$$

Each $\lambda S \in \lambda\mathit{Cube}$ is a type assignment system The judgements have a form $\Gamma \vdash A : B$ where Γ is a prebasis, and A and B are prephases. The derivation rules for each λS is the followings.

Initial rule: $\dfrac{}{\vdash * : \Box}$

Tautology: $\dfrac{\Gamma \vdash A : s}{\Gamma, x : A \vdash x : A}$ $(s \in \{*, \Box\})$

Exchange: $\dfrac{\Gamma \vdash A : B}{\Gamma' \vdash A : B}$ $(\Gamma \simeq \Gamma')$

Weakening: $\dfrac{\Gamma \vdash A : B \qquad \Gamma' \vdash A' : B'}{\Gamma'' \vdash A' : B'}$ (Γ'' is a merge of Γ and Γ')

Quantification: $\dfrac{\Gamma, x : A \vdash B : s \qquad \Gamma \vdash A : s'}{\Gamma \vdash \Pi x : A.B : s}$ $(\langle s', s\rangle \in |S|)$

Abstraction: $\dfrac{\Gamma, x : A \vdash M : B \qquad \Gamma \vdash \Pi x : A.B : s}{\Gamma \vdash \lambda x : A.M : \Pi x : A.B}$

Application: $\dfrac{\Gamma \vdash M : \Pi x : A.B \qquad \Gamma \vdash N : A}{\Gamma \vdash MN : B[N/x]}$

Beta-Conversion: $\dfrac{\Gamma \vdash A : B \qquad \Gamma \vdash B : s \qquad \Gamma \vdash B' : s}{\Gamma \vdash A : B'}$

($s \in \{*, \Box\}$, B is β-equivalent to B' by one-step β-conversion.)

Only the rule of quantification depends on S. All the other rules are common.

Lemma 2.1.6 *Strong normalisability, confluency and subject reduction property on β-reduction hold for each λS.*

Proof. Shown in the literature [Barendregt]. □

Definition 2.1.7 (Term, Type, Formula, Predicate, Kind, Phrase, Basis)
A prephrase A is a ***kind*** iff some λS derives $\Gamma \vdash A : \Box$.
A prephrase T is a *type* iff some λS derives $\Gamma \vdash T : *$.
A type is also called a *formula.*

A prephrase P is a *predicate* iff some λS derives $\Gamma \vdash P : A$ for some kind A.
A prephrase M is a *term* iff some λS derives $\Gamma \vdash M : T$ for some type T.
A prephrase is a *phrase* iff it is a term, a predicate, a kind, or $\Box$.
A prebasis Γ is a *basis* iff some λS derives $\Gamma \vdash A : B$.

Notation 2.1.8 A notation $A \to B$ is an abbreviation of $\Pi x : A.\, B$, where x is fresh. We write $\lambda x^T.e$ and $\Pi x^T.P$ for $\lambda x{:}T.e$ and $\Pi x{:}T.P$. We write $\lambda xy.e$ for $\lambda x^T.\lambda y^U.e$, and $\Pi xy.P$ for $\Pi x^T.\Pi y^U.P$, and so forth.

Notation 2.1.9 We write $e =_\lambda f$ for β-equality, because we sometimes use β as a meta-variable.

2.2 Lambda Cube as a Logical System

We regard λS's as logical systems as well as type assignment systems. Thus, we define the notions of axioms, theorems and so forth over the systems of λS's.

Definition 2.2.1 (Infinite Basis) Let $\Gamma = \{x_i : T_i\}_{i<\alpha}$ be an infinite set indexed by ordinal numbers less than an infinite ordinal number α, where for each $i < \alpha$, x_i is a variable and T_i is a phrase. The indexed set Γ is an *infinite* λS*-bases*, or an *infinite bases* of λS, iff for each $i < \alpha$,
– $x_i \notin \{x_j \mid j < i\}$, and
– there is a finite sequence of ordinal numbers $i_1, i_2, ..., i_n$ such that $0 \leqq i_1 < i_2 < \cdots < i_n < i$ and λS derives $x_{i_1} : T_{i_1}, \ ..., \ x_{i_n} : T_{i_n} \vdash T_i : s$ where s is $*$ or $\Box$.
For a bases or infinite basis Γ and an infinite bases Γ, the relation of subsequence $\Gamma \leq \Gamma'$ is defined in the similar way to that for finite sequences.

Definition 2.2.2 (Derivation of Infinite Basis) For an infinite λS-basis Γ, we say that λS derives $\Gamma \vdash A : B$ when there is a basis Γ' such that $\Gamma' \leq \Gamma$ and λS derives $\Gamma' \vdash$.

Definition 2.2.3 (Axiom Set, Theorem) Let $A \equiv \{a_i : A_i\}_{i<n+n'}$ be an finite or infinite bases of λS such that λS derives $\vdash A_i : *$ for each i. Then, we sometimes regard A as an *axiom set*. If λS derives $\Gamma \vdash P : *$ and $\Gamma, A \vdash M : P$ for some phrase M, then we call P a *theorem* of A, and we say that A *proves* P in λS.

Notation 2.2.4 We use the following notations when we regard these types as formulae.

$$P \supset Q :\equiv P \to Q, \quad \forall x^T.P :\equiv \forall x : T.\, P :\equiv \Pi x : T.\, P,$$
$$P \wedge Q :\equiv \forall X^*.(P \supset Q \supset X) \supset X.$$
$$x = y :\equiv \forall X^{T \to T \to *}.\, Xx \supset Xx, \quad \mathcal{E}q_T :\equiv \lambda x^T y^T.\, x = y$$

Let P and Q be predicates of kind $A \equiv \Pi x_1^{T_1} ... x_n^{T_n}.*$, then

$$P \sqsubseteq Q :\equiv \forall x_1 ... x_n.\ Px_1 ... x_n \supset Qx_1 ... x_n,$$
$$P \cong Q :\equiv P \sqsubseteq Q \wedge Q \sqsubseteq P, \quad \cong_A :\equiv \lambda X^A Y^A.X \cong Y.$$

3 Axioms of Parametricity

3.1 Predicate Logic Over Terms of Each System

We define $\lambda\hat{S}$ for each λS. Each system $\lambda\hat{S}$ play the role of second order predicate logic over terms of λS. Second order universal quantifying appears in the definition of parametricity. Hence we discuss the formal theory of parametricity of the system λS on $\lambda\hat{S}$ regarded as the logical system. The formal definition is the follows.

Definition 3.1.1 A map $S \mapsto \hat{S}$ is a function of *Cube* into *Cube* such that $|\hat{S}| = |\mathrm{P2}| \cup |S|$. Thus, $\hat{S} = \mathrm{P2}$ for $S \in \{ \rightarrow, \mathrm{P}, 2, \mathrm{P2} \}$ and $\hat{S} = \mathrm{P}\omega$ for $S \in \{ \underline{\omega}, \mathrm{P}\underline{\omega}, \omega, \mathrm{P}\omega \}$

We will define the axioms of parametricity for terms of λS as formulae of $\lambda\hat{S}$. The conformity relation $\langle\!\langle - \rangle\!\rangle^{\{\}}$ played the essential role in the theory of axiomatising parametricity for System F in the literature [T]. We would like to define this conformity relation for each system of λ-cube. As for System F, the relation $\langle\!\langle T \rangle\!\rangle^{\{\Theta\}}$ is defined by induction on T. There are only types appear as subexpressions of a type. Therefore it is sufficient to define $\langle\!\langle T \rangle\!\rangle^{\{\Theta\}}$ only for types T. But, in the systems of λ-cube except for $\lambda \rightarrow$ and $\lambda 2$, there are many objects of other levels appears as subexpressions of a type. Therefore we have to define $\langle\!\langle P \rangle\!\rangle^{\{\Theta\}}$ for P which may not be a type. In order to define that, we define a notation $(\!| P : \alpha |\!)^{\{\Theta\}}$ and call it the kind of conformity notation.

3.2 Kind of Conformity Relation

Definition 3.2.1 (Double Assignment) A *double assignment* is a set of expressions of the form $(e, f)/x$ which satisfies the followings:
– Each component $(e, f)/x \in \Theta$ consists of a variable x and two phrases e and f.
– For each variable x, there is at most one component $(e, f)/x \in \Theta$.
The component $(e, f)/x \in \Theta$ means that Θ assigns the pair of phrases (e, f) to the variable x.

Definition 3.2.2 (Domain of a Double Assignment) Let Θ be a double assignment such that $\Theta = \{(e_1, f_1)/x_1, (e_2, f_2)/x_2, ..., (e_n, f_n)/x_n\}$. Then $dom(\Theta)$ is the set of variable such that $dom(\Theta) = \{x_1, x_2, ..., x_n\}$. It is called the *domain of the double assignment* Θ.

Definition 3.2.3 Let Θ be a double assignment such that $\Theta = \{(e_1, f_1)/x_1, (e_2, f_2)/x_2, ..., (e_n, f_n)/x_n\}$. Then Θ^l and Θ^r are the substitutions of the double assignment which are defined as $\Theta^l = \{e_1/x_1, e_2/x_2, ..., e_n/x_n\}$ and $\Theta^r = \{f_1/x_1, f_2/x_2, ..., f_n/x_n\}$.

Definition 3.2.4 (Double Assignment for a Declaration) Let Γ and Γ' be declarations. Let Θ be a double assignment. Then the double assignment Θ is a *double assignment for* Γ *under* Γ' iff Θ satisfies the followings:

- $dom(\Theta) \subset dom(\Gamma)$.
- For each $x : T \in \Gamma$, either $x \in dom(\Theta)$ or $\Gamma' \vdash x : T$.
- For each component $(e, f)/x \in \Theta$, if $\Gamma \vdash x : T$, then $\Gamma' \vdash e : T\Theta^l$ and $\Gamma' \vdash f : T\Theta^r$.

If Θ is a double assignment for Γ under Γ', then we write $\Gamma' \vdash \Gamma \rhd \Theta$.

Definition 3.2.5 (Kind of Conformity Relation) Let P be a predicate of a kind α under some Γ. Let Θ be a double assignment. Then the *kind of conformity relation* of $P : \alpha$ is defined as below:

$$(\!|P : *|\!)^{\{\Theta\}} :\equiv P\Theta^l \to P\Theta^r \to *$$

$$(\!|P : \Pi x^T.\alpha|\!)^{\{\Theta\}} :\equiv \Pi x_1{}^{T\Theta^l} x_2{}^{T\Theta^r}.(\!|Px : \alpha|\!)^{\{\Theta,(x_1,x_2)/x\}}$$

for some type T under Γ.

$$(\!|P : \Pi X^\beta.\alpha|\!)^{\{\Theta\}} :\equiv$$

$$\Pi X_1{}^{\beta\Theta^l} X_2{}^{\beta\Theta^r}.\Pi Y : (\!|X : \beta|\!)^{\{\Theta,(X_1,X_2)/X\}}.(\!|PX : \alpha|\!)^{\{\Theta,(X_1,X_2)/X\}}$$

for some kind β under Γ.

Lemma 3.2.6 *Let Γ and Γ' be declarations. Let Θ be a double assignment such that $\Gamma \vdash \Gamma \rhd \Theta$. Let α be a kind and P be a predicate such that $\Gamma \vdash P : \alpha$.*

Then, it is derivable that $\Gamma' \vdash (\!|P : \alpha|\!)^{\{\Theta\}} : *$.

Proof. Induction on α. □

Proposition 3.2.7 *If $\alpha =_\lambda \beta$, then $(\!|P : \alpha|\!)^{\{\Theta\}} =_\lambda (\!|P : \beta|\!)^{\{\Theta\}}$.*

Proof. Easy. □

3.3 Conformity Relation

Definition 3.3.1 (Multiple Assignment) A *multiple assignment* is a set of expressions of the form $(e, f)/x$ or of the form $(e, f, g)/x$ which satisfies the followings:

- Each component is either of the form $(e, f)/x$ which consists of a variable x and two phrases e and f, or of the form $(e, f, g)/x$ which consists of a variable x and three phrases e, f and g.
- For each variable x, there is at most one component $(e, f)/x$ or $(e, f, g)/x \in \Theta$.

The component $(e, f)/x \in \Theta$, or $(e, f, g)/x \in \Theta$, means that Θ assigns the pair of phrases (e, f), or (e, f, g) respectively, to the variable x.

Definition 3.3.2 For a multiple assignment Θ, the double assignment Θ^{lr} is defined as $(e, f)/x \in \Theta^{lr}$ iff $(e, f)/x \in \Theta$, or $(e, f, g)/x \in \Theta$ for some phrase g.

In other words, for a multiple assignment Θ, the double assignment Θ^{lr} is made by omitting the third phrase in each component of Θ with three phrases.

Definition 3.3.3 Let Θ be a multiple assignment. Then the substitutions Θ^l and Θ^r are defined as $\Theta^l = (\Theta^{lr})^l$ and $\Theta^l = (\Theta^{lr})^r$.

Definition 3.3.4 (Multiple Assignment for a Declaration).

Let Γ and Γ' be declarations. Let Θ be a multiple assignment. Then the Multiple assignment Θ is a *multiple assignment for Γ under Γ'* iff Θ satisfies the followings:
- $dom(\Theta) \subset dom(\Gamma)$.
- For each $x : T \in \Gamma$, either $x \in dom(\Theta)$ or $\Gamma' \vdash x : T$.
- For each component $(e, f)/x \in \Theta$, if $\Gamma \vdash x : T$, then the following is derivable:
$$\Gamma' \vdash T\Theta^l : *, \quad \Gamma' \vdash T\Theta^r : *,$$
$$\Gamma' \vdash e : T\Theta^l, \quad \Gamma' \vdash f : T\Theta^r.$$
- For each component $(e, f, g)/x \in \Theta$, if $\Gamma \vdash x : T$, then the following is derivable:
$$\Gamma' \vdash T\Theta^l : \Box, \quad \Gamma' \vdash T\Theta^r : \Box,$$
$$\Gamma' \vdash e : T\Theta^l, \quad \Gamma' \vdash f : T\Theta^r,$$
and $\Gamma' \vdash g : (\!|x : T|\!)^{\{\Theta\}}$.
If Θ is a double assignment for Γ under Γ', then we write $\Gamma' \vdash \Gamma \rhd \Theta$.

Definition 3.3.5 (Conformity Relation) Let P be a predicate under some Γ. Let Θ be a multiple assignment. Then the *kind of conformity relation* of P is defined as below:
- For a variable X^A,
$$\langle\!\langle X \rangle\!\rangle^{\{\Theta\}} :\equiv \cong_A \quad \text{where } X \notin dom(\Theta)$$
$$\langle\!\langle X \rangle\!\rangle^{\{\Theta\}} :\equiv Q \quad \text{where } (P_1, P_2, Q)/X \in dom(\Theta)$$
- For a type T and a term e under Γ,
$$\langle\!\langle \Pi x : T.P \rangle\!\rangle^{\{\Theta\}} :\equiv \lambda y^{(\Pi x:T.P)\Theta^l} z^{(\Pi x:T.P)\Theta^l}.$$
$$\forall x_1^{T\Theta^l} x_2^{T\Theta^r}.\langle\!\langle T \rangle\!\rangle^{\{\Theta\}} x_1 x_2 \supset \langle\!\langle T \rangle\!\rangle^{\{\Theta,(x_1,x_2)/x\}} (y x_1)(z x_2)$$
$$\langle\!\langle \lambda x : T.P \rangle\!\rangle^{\{\Theta\}} :\equiv \lambda x_1^{T\Theta^l} x_2^{T\Theta^r}.\langle\!\langle Px \rangle\!\rangle^{\{\Theta,(x_1,x_2)/x\}}$$
$$\langle\!\langle Pe \rangle\!\rangle^{\{\Theta\}} :\equiv \langle\!\langle P \rangle\!\rangle^{\{\Theta\}} (e\Theta^l)(e\Theta^r)$$
- For a kind α and a predicate Q under Γ,
$$\langle\!\langle \Pi X : \alpha.P \rangle\!\rangle^{\{\Theta\}} :\equiv$$
$$\forall X_1^{\alpha\Theta^l} X_2^{\alpha\Theta^r}.\forall Y : (\!|X : \alpha|\!)^{\{\Theta^{lr},(X_1,X_2)/X\}}.\langle\!\langle P \rangle\!\rangle^{\{\Theta,(X_1,X_2,Y)/X\}}$$
$$\langle\!\langle \lambda X : \alpha.P \rangle\!\rangle^{\{\Theta\}} :\equiv$$
$$\lambda X_1^{\alpha\Theta^l} X_2^{\alpha\Theta^r}.\lambda Y : (\!|X : \alpha|\!)^{\{\Theta^{lr},(X_1,X_2)/X\}}.\langle\!\langle P \rangle\!\rangle^{\{\Theta,(X_1,X_2,Y)/X\}}$$
$$\langle\!\langle PQ \rangle\!\rangle^{\{\Theta\}} :\equiv \langle\!\langle P \rangle\!\rangle^{\{\Theta\}} (Q\Theta^l)(Q\Theta^r)\langle\!\langle Q \rangle\!\rangle^{\{\Theta\}}$$

Lemma 3.3.6 *Let Γ and Γ' be declarations of λS. Let Θ be a multiple assignment such that λS derives $\Gamma \vdash \Gamma \rhd \Theta$. Let α be a kind and P be a predicate such that λS derives $\Gamma \vdash P : \alpha$.*
Then $\lambda\hat{S}$ derives $\Gamma' \vdash \langle\!\langle P \rangle\!\rangle^{\{\Theta\}} : (\!|P : \alpha|\!)^{\{\Theta^{lr}\}}$.

Proof. Induction on P. □

Notation 3.3.7 If the multiple assignment is empty, then $(\!|P : \alpha|\!) :\equiv (\!|P : \alpha|\!)^{\{\}}$ and $\langle\!\langle P \rangle\!\rangle :\equiv \langle\!\langle P \rangle\!\rangle^{\{\}}$. Suppose that $\Gamma \vdash T : *$, $\Gamma \vdash e : T$ and $\Gamma \vdash f : T$. Then $\langle\!\langle T \rangle\!\rangle ef$ is a formula under Γ. We read this formula $\langle\!\langle T \rangle\!\rangle ef$ as e is *conforming* to f, or e and f is conforming to each other.

Proposition 3.3.8 *If $P =_\lambda Q$, then $\langle\!\langle P \rangle\!\rangle^{\{\Theta\}} =_\lambda \langle\!\langle Q \rangle\!\rangle^{\{\Theta\}}$.*

Proof. Easy. □

Remark 3.3.9 If P is a $\lambda 2$-type, and all the phrases in Θ are $\lambda 2$-phrases, then the conformity relation $\langle\langle P \rangle\rangle^{\{\Theta\}}$ is equivalent to that in the theory of axiomatising parametricity for System F. Thus, this definition of conformity is an extension of that in the theory of axiomatising parametricity for System F.

Definition 1. (Axiom of Parametricity) Let T be a type under basis Γ in λS. Then, *the axiom of parametricity for* T is defined as: $\mathbf{Par}(T) :\equiv \mathcal{E}q_T \cong \langle\langle T \rangle\rangle$. *The set of axioms of parametricity for* λS is defined as:

$$\mathbf{Par}_{\lambda S} :\equiv \{\forall \Gamma.\mathbf{Par}(T) |\ \lambda S \text{ derives } \Gamma \vdash T : * \ \}.$$

3.4 Consistency

Theorem 3.4.1 (Consistency) $\mathbf{Par}_{\lambda S}$ *does not prove* $\forall X^*.\forall x^X y^X.x = y$. *which means corruption of calculation.*

Proof. The proof is done by using the relativisation which is similar to that in [T]. The relativisation reduces the consistency of $\mathbf{Par}_{\lambda S}$ to normalisability of $\lambda \hat{S}$ which is shown in Lemma 2.1.6. □

Remark 3.4.2 The formula $\forall X^*.\forall x^X y^X.x = y$ means corruption of calculation.

4 Category Theory

We will show an application of the theory of parametricity to a category theory in formal system. First we will interpret category theory into $\lambda P\omega$. Then we show an internal theorem, which is adjunction functor theorem.

4.1 Basic Notations

Definition 4.1.1 (Category, Object) A kind is regarded as a *category*. A predicate P of kind A is regarded as an *object* of the category A.

Notation 4.1.2 Let M be a type or a predicate, and Γ be a prebasis such that $\Gamma \equiv x_1 : T_1, ..., x_n : T_n$. Then,

$$\Pi\Gamma.P \equiv \forall\Gamma.P :\equiv \Pi x_1^{T_1}...x_n^{T_n}.\ P,$$
$$\lambda\Gamma.M :\equiv \lambda x_1^{T_1}...x_n^{T_n}.\ M,$$
$$M\Gamma :\equiv Mx_1...x_n.$$

Definition 4.1.3 (Hom-set, Arrow, Identity, Composition) Let Γ be a prebasis and C be a category such that $C \equiv \Pi\Gamma.*$. Let A and B be objects of the category C. Then the type

$$A \hat{\to} B :\equiv \Pi\Gamma.\ A\Gamma \to B\Gamma\ (\equiv A \sqsubseteq B)$$

is regarded as the *hom-set* $Hom_C(A, B)$, and a term f of type $A \hat{\to} B$ is an *arrow* of $Hom_C(A, B)$. For an object A of a category C, the arrow

$$\mathbf{id}_A :\equiv \lambda\Gamma.\lambda x^{A\Gamma}.\ x$$

is regarded as the *identity arrow* of P. For arrows $f : A \hat{\to} A'$ and $g : A' \hat{\to} A''$, the arrow

$$g \circ_C f :\equiv \lambda\Gamma.\lambda x^{A\Gamma}.g\Gamma(f\Gamma x)$$

is regarded as the *composition* of f and g. We sometimes omit the type index and write only $\circ$ for $\circ_C$.

Proposition 4.1.4 *For arrows $f : A_1 \hat{\to} A_2$, $g : A_2 \hat{\to} A_3$, and $h : A_3 \hat{\to} A_4$, we have $\mathbf{id}_{A_2} \circ f =_\lambda f \circ \mathbf{id}_{A_1}$, and $h \circ (g \circ f) =_\lambda (h \circ g) \circ f$.*

Notation 4.1.5 $e \circ (f, g) :\equiv \lambda xy.e(fx)(gy)$. This notation is similar to $f \circ g \equiv \lambda x.f(gx)$.

Definition 4.1.6 (Functor) Let C, D be categories. Let F_{type} be a predicate of kind $C \to D$, F_{pred} be a predicate of kind $(\!|F_{type} : C \to D|\!)$, and F_{func} be a term of type $\Pi XY : C.(X \hat{\to} Y) \to F_{type}X \hat{\to} F_{type}Y$. Then, the internal formula that

the triple $(F_{type}, F_{pred}, F_{func})$ is a *functor* of C into D

denotes the conjunctions of the following formulae:

- Identity Expansion: $\forall X : C.\ F_{pred}XX\mathcal{E}q_X \cong \mathcal{E}q_{F_{type}X}$
- Functoriality:

$$\forall XX'YY' : C.\ \forall x : X \hat{\to} X'.\forall y : Y \hat{\to} Y'.$$
$$\forall \phi : (\!|X : C|\!)^{\{(X,X')/X\}}.\ \forall \psi : (\!|Y : C|\!)^{\{(Y,Y')/Y\}}.$$
$$\phi \sqsubseteq \psi \circ (x, y) \supset F_{pred}XY\phi \sqsubseteq F_{pred}X'Y'(\psi \circ (F_{func}XX'x, F_{func}YY'y))$$

Notation 4.1.7 Let F be a triple $(F_{type}, F_{pred}, F_{func})$ which is a functor of C into D. Then, we write simply as follows:

$F(A) :\equiv F_{type}A$ for an object $A : C$,
$F(\Phi) :\equiv F_{pred}AB\Phi$ for a predicate $\Phi : (\!|X : C|\!)^{\{(A,B)/X\}}$
$F(f) :\equiv F_{type}ABf$ for an arrow $f : A \hat{\to} B$

Then identity expansion and functoriality is written as follows:

- Identity Expansion: $\forall X.\ F(\mathcal{E}q_X) \cong \mathcal{E}q_{F(X)}$
- Functoriality: $\forall XX'YY'xy\phi\psi.\ \phi \sqsubseteq \psi \circ (x, y) \supset F(\phi) \sqsubseteq F(\psi) \circ (F(x), F(y))$

Proposition 4.1.8 *Let F be a functor of C into D. Let A, A' and A'' be objects of C. Let f and g be arrows such that $f : A \dot{\to} A'$ and $g : A' \dot{\to} A''$. Then, the system $\lambda\omega$ or $\lambda P\omega$ proves that*

$$F(\mathbf{id}_A) = \mathbf{id}_{F(A)} \quad \textit{and} \quad F(g \circ_C f) = F(g) \circ_D F(f),$$

We can describe the notion of adjunction in $\lambda P\omega$, that is, we can describe a formula of $\lambda P\omega$ which states that a functor F of a category C into a category D is a left adjoint of a functor G of a category D into a category C.

4.2 Adjunction

Notation 4.2.1 Let $C \equiv \Pi\Gamma.*$ be a category. and A be an object of C. Then,

$\Pi' x : T.A :\equiv \lambda\Gamma.\Pi x^T.A\Gamma$
$T \dot{\to} A :\equiv \Pi' x^T.A$ where X is fresh.

Note that both $\Pi' x^T.A$ and $T \dot{\to} A$ are also objects of C.

Remark 4.2.2 Let F be a functor of C into C'. Then $\Pi' X^C.F(X)$ plays the role of limit.

Definition 4.2.3 (Preserving Limits) When we say that a functor F of C into C' preserves limits, we mean a formula which states the following condition internally:

The following two arrows are isomorphisms and natural with respect to A and T where T is a type.

$$j_{T,A} :\equiv \lambda z^{F(\Pi' x.Ax)} x^T.F(\lambda y.yx)z : F(\Pi' x^T.Ax) \hat{\to} \Pi' x^T.F(Ax)$$
$$j'_A :\equiv \lambda z^{F(\Pi' X.AX)} X^C.F(\lambda y.yX)z : F(\Pi' X^{C'}.AX) \hat{\to} \Pi' X^{C'}.F(AX).$$

Definition 4.2.4 For a functor $F = (F_{type}, F_{pred}, F_{func})$ of C into C', the triple $F^\dashv = (F^\dashv_{type}, F^\dashv_{pred}, F^\dashv_{func})$ defined as follows:

$$F^\dashv_{type} :\equiv \lambda X^{C'}.\ \Pi' Y^C.(X \hat{\to} F(Y)) \dot{\to} Y$$
$$F^\dashv_{pred} :\equiv \langle\!\langle \lambda X^{C'}.\ \Pi' Y^C.(X \hat{\to} ZY) \dot{\to} Y \rangle\!\rangle^{\{(C\to C', C\to C', F_{pred})/Z\}}$$
$$F^\dashv_{func} :\equiv \lambda X^C X'^C.\lambda x^{X \hat{\to} Y}.\lambda \Gamma.\lambda y^{F(X)\Gamma}.y\lambda \Gamma' \lambda z^{X\Gamma' \to F(Y)\Gamma'}.z \circ x$$

where $C = \Pi\Gamma.*$ and $C' = \Pi\Gamma'.*$.

Proposition 4.2.5 $\mathbf{Par}_{\lambda\omega}$ *proves that for each functor F of C into C', the triple $F^\dashv$ is a functor of C' to C.*

Theorem 4.2.6 (Adjoint Functor Theorem) *The axioms* $\mathbf{Par}_{\lambda P\omega}$ *proves the formula which states the following assertion.*

If a functor F preserves limits, then $F^\dashv$ is a left adjoint of F.

Remark 4.2.7 If F preserves limits, then $F^\dashv X$ is isomorphic to $\Pi Y.F(X \to Y) \to Y$, which is close to $\neg F(\neg X)$ logically, as is mentioned in the introduction of literature [Hasegawa3].

4.3 Examples

Example 4.3.1 (Exponentiation and Product) Let A be a type. Let $(A \to -)$ be a functor of $*$ into $*$ which maps an object B to $A \to B$. The formal definition is as follows:

$$(A \to -)_{type} :\equiv \lambda X^*.A \to X$$
$$(A \to -)_{pred} :\equiv \langle\!\langle (A \to -)_{type} \rangle\!\rangle$$
$$(A \to -)_{func} :\equiv \lambda X^* Y^* x^{X\to Y} y^{A\to X}.x \circ y$$

Then, this functor $(A \to -)$ preserves limits under $\mathbf{Par}_{\lambda\omega}$. Therefore, the left adjoint $(A \to -)^\dashv$ is the functor which maps an object B to $\Pi X : *.(B \to A \to X) \to X$, which is isomorphic to $A \times B \equiv \Pi X : *.(A \to B \to X) \to X$. Thus, the axiom set $\mathbf{Par}_{\lambda\omega}$ proves that the functor $(A \times -)$ is a left adjoint of $(A \to -)$.

Example 4.3.2 (Existential Quantifier) Let T be a type. Let K be a functor of $*$ into $T \to *$ which maps an object P to $\lambda x^T.P$. Then, this functor K preserves limits under $\mathbf{Par}_{\lambda \mathrm{P}\omega}$. Therefore, the left adjoint $K^{\dashv}$ is the functor which maps an object $P^{T \to *}$ to $\Pi X^* : (P \hat{\to} \lambda x^T.X) \to X$, which is identical to $\forall X^*.(\forall x^T.Px \supset X) \supset X$. This is the standard encoding of $\exists x^T.Px$. Thus, the axiom set $\mathbf{Par}_{\lambda \mathrm{P}\omega}$ proves that the existential quantifier is a left adjoint of K.

Example 4.3.3 (Universal Quantifier) Let T be a type. Let $\forall$ be a functor of $T \to *$ into $*$ which maps an object P to $\Pi x^T.Px \equiv \forall x^T.Px$. Then, this functor $\forall$ preserves limits under $\mathbf{Par}_{\lambda \mathrm{P}\omega}$. Therefore, the left adjoint $\forall^{\dashv}$ is the functor which maps an object P to $\Pi X^{T \to *} : (P \to \Pi x^T.Xx) \dot{\to} X$. We can prove that this $\forall^{\dashv} P$ is isomorphic to KP under $\mathbf{Par}_{\lambda \mathrm{P}\omega}$. Thus, the axiom set $\mathbf{Par}_{\lambda \mathrm{P}\omega}$ proves that the universal quantifier is a right adjoint of K.

5 Conclusion

5.1 Main result

The extend the theory of parametricity in lambda cube, that is, we defined the theory of parametricity for each of eight system in lambda cube. The definitions of them are uniform, and the theory of parametricity for $\lambda 2$ is equivalent to that in the precedent works [PA, T]. This fact certificate our extension.

In some systems in lambda cube, We can deal with several notions which come from category theory. Especially, the system $\lambda\omega$ can deal with functors and adjunction. The theory of parametricity for $\lambda\omega$ proves the adjoint theorem. This is our main result. The theory of parametricity for $\lambda\omega$ is a theory in the system $\lambda \mathrm{P}\omega$, which is equivalent to the system of calculus of construction. Therefore, we paraphrase our main result as the following: We formalised the notions of functors and adjunction in calculus of construction, and proved the adjoint theorem by the theory of parametricity.

5.2 Future work

We have many other properties which we did not discuss in this paper. Poll and Zwanenburg proposed to put a useful property *ParQuot* as an axiom in his paper [PZ]. The formula *ParQuot* states that for each type X and for each equivalence relation R over X, there is a type Q which denotes the quotient domain obtained by the domain X divided by R. Hasegawa Masahito has the conjecture that $\mathbf{Par}\mathrm{P}\omega$ proves the axiom *ParQuot*.

As for category theory, our theory can give the interpretation of only the restricted categories. The class of such categories is called *Uni* by Hasegawa Ryu [Hasegawa3]. It seems attractive to extend the class of categories which can be interpreted. The systems of λ-Cube has only four levels of objects, which are of terms, of types, of kinds, and of $\Box$. In order to extend the class of categories, we need to extend our type theory with more levels of objects.

Acknowledgement

I am greatful to Sato Masahiko for his total support. I am very debt to Hasegawa Ryu and Hasegawa Masahito for their comments and encouragements.

References

[ACC] Abadí, M., Cardelli, L., & Curien, P.-L.: Formal parametric polymorphism, *Theoretical Computer Science*, **121**, pp9–58, (1993).

[Barendregt] Barendregt, H.: Lambda Calculi with Types, *Handbook of Logic in Computer Science,* Vol. 2, pages 117–309, Oxford Univ. Press, 1992, edited by S. Abramsky et al.

[BAC] Bellucci, R., Abadí, M., & Curien, P.-L., A Model for Formal Parametric Polymorphism: A PER Interpretation for system R, *Typed Lambda Calculi and Applications, 1995,* Lecture Notes in Computer Science, 902, pages 32–46 (1995).

[Girard] Girard J.-Y. et al. *Proofs and Types,* Cambridge Univ. Press, 1989.

[Hasegawa1] Hasegawa, R.: Parametricity of extensionally collapsed term models of polymorphism and their categorical properties, *Proceedings of Theoretical Aspects of Computer Science,* Lecture Notes In Computer Science 526, pages 495–512, Springer-Verlag, 1991, edited by T. Ito et al.

[Hasegawa2] Hasegawa, R.: Categorical data types in parametric polymorphism. *Mathematical Structures in Computer Science,* 4:71–109, 1994.

[Hasegawa3] Hasegawa, R.: Relational limits in general polymorphism. *Publications of the Research Institute for Mathematical Sciences,* Vol. 30 (1994), 535–576.

[HS] Huet, G. & Saïbi, A.: Constructive Category Theory, CLICS-TYPES BRA meeting, Jan. 1995, Final version published in: *Proof, Language and Interaction. Essays in Honour of Robin Milner*, Ed. Gordon Plotkin, Colin Stirling and Mads Tofte. MIT Press, 2000.

[PA] G. Plotkin, G., & Abadí, M.: A Logic for parametric polymorphism, *Typed Lambda Calculi and Applications, 1993,* edited by M. Bezem et al., Lecture Notes in Computer Science, 664, pages 361–375, Springer Verlag, 1993.

[PP] Pfenning, F. & Paulin-Mohring, C.: Inductively defined types in the calculus of constructions. *Mathematical foundations of programming semantics* (New Orleans, LA, 1989), 209–228, Lecture Notes in Comput. Sci., 442, Springer, Berlin, 1990.

[PZ] Pool, Erik & Zwanenburg, Jan: A Logic for Abstract Data Types as Existential Types. *Typed Lambda Calculus, 1999*, edited by GIrard, J.-Y., Lecture Notes in Computer Science, 1581, pages 310–324, Springer Verlag, 1999.

[Reynolds] Reynolds, J. C.: Types, abstraction and parametric polymorphism, R. E. A. Mason, editor, *Information Processing 83,* pages 513–523, Elsevier Science Publishers B. V., 1983.

[T] Takeuti, I.: An Axiomatic System of Parametricity, *Typed Lambda Calculi and Applications, 1997,* Lecture Notes in Computer Science, 1210, pages 354–372 (1997). & *Fundamenta Informatica*, **33**, pp397–432, (1998).

This article was processed using the LaTeX macro package with LLNCS style